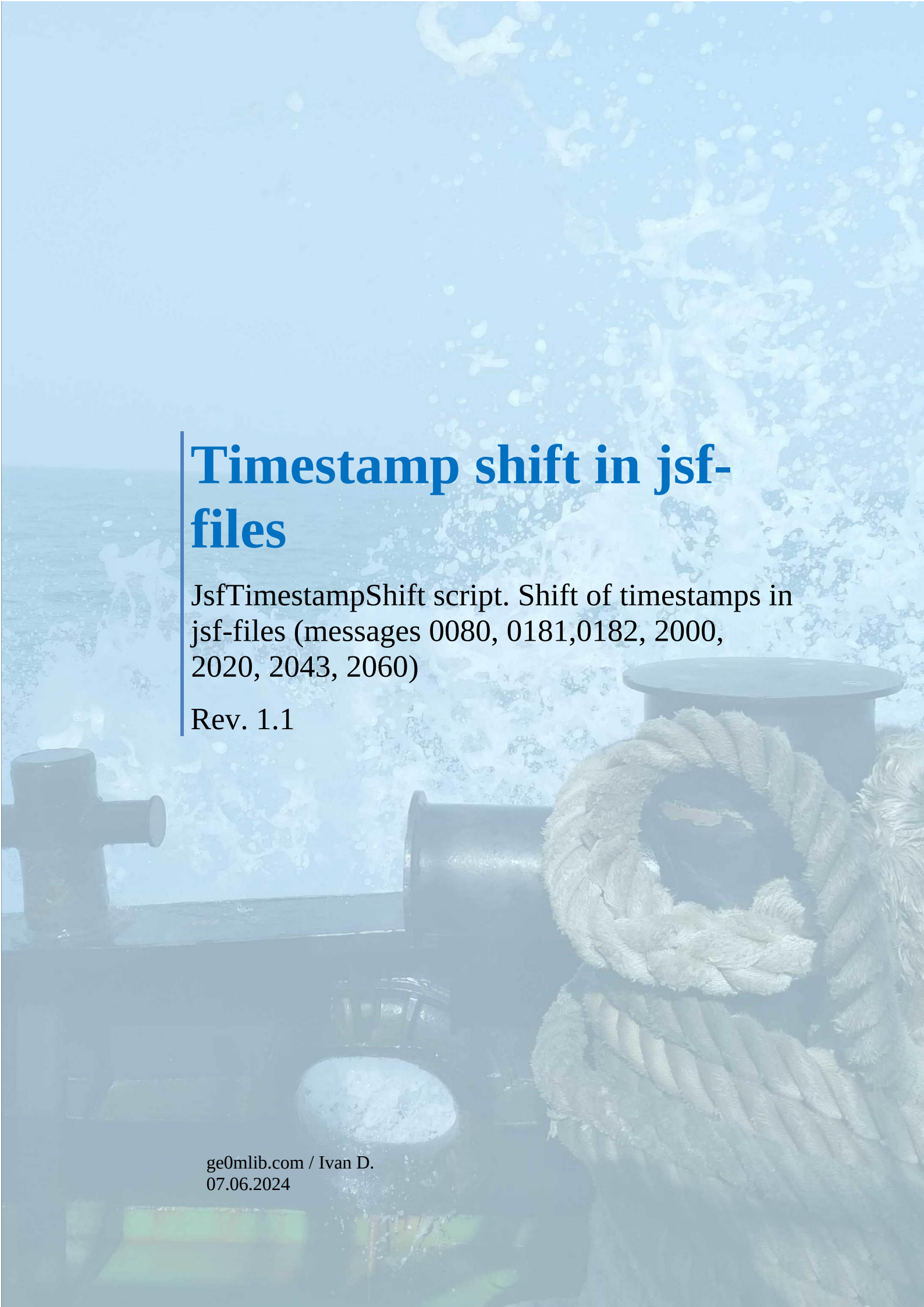

# Timestamp shift in jsf-files

JsfTimestampShift script. Shift of timestamps in jsf-files (messages 0080, 0181,0182, 2000, 2020, 2043, 2060)

Rev. 1.1

ge0mlib.com / Ivan D.
07.06.2024

## *Contents*

## *History of changes*

| Version | Date | Description of changes |
|---|---|---|
| 1.0 | 2023-08-27 | Document created // Ivan D. |
| 1.1 | 2024-06-07 | Edited the text // Ivan D. |
| | | |

## *Figures list*



## *Tables list*

# 1 Introduction

The script performs **timestamps shift in a jsf-file for a defined value to the nearest milliseconds**. Each jsf-file contains a sequence of “recordings” or “messages” received from different hardware sensors and written to the file in order they were received. Logging of the jsf-files, for the modification of which the script was created, was performed using the SSS ET4205; the files contain the following message types 0040, 0080, 0181, 0182, 2000, 2020, 2043, 2060. Timestamp shifts are performed for messages 0080, 2000, 2020, 2043, 2060.

The script performs batch processing of jsf-files located in a specified folder. The script requires a text file containing the file name and time shift value for each jsf-file. The window with the script operation log is shown in *Figure 1.1*; during processing, the following information is displayed in the window (shown by arrows in *Figure 1.1*): the name of the processed file, the type name of the processed message, the file processing time.

To run the script, MatLab 2018a (or later) and the ge0mlib library are needed to be installed.

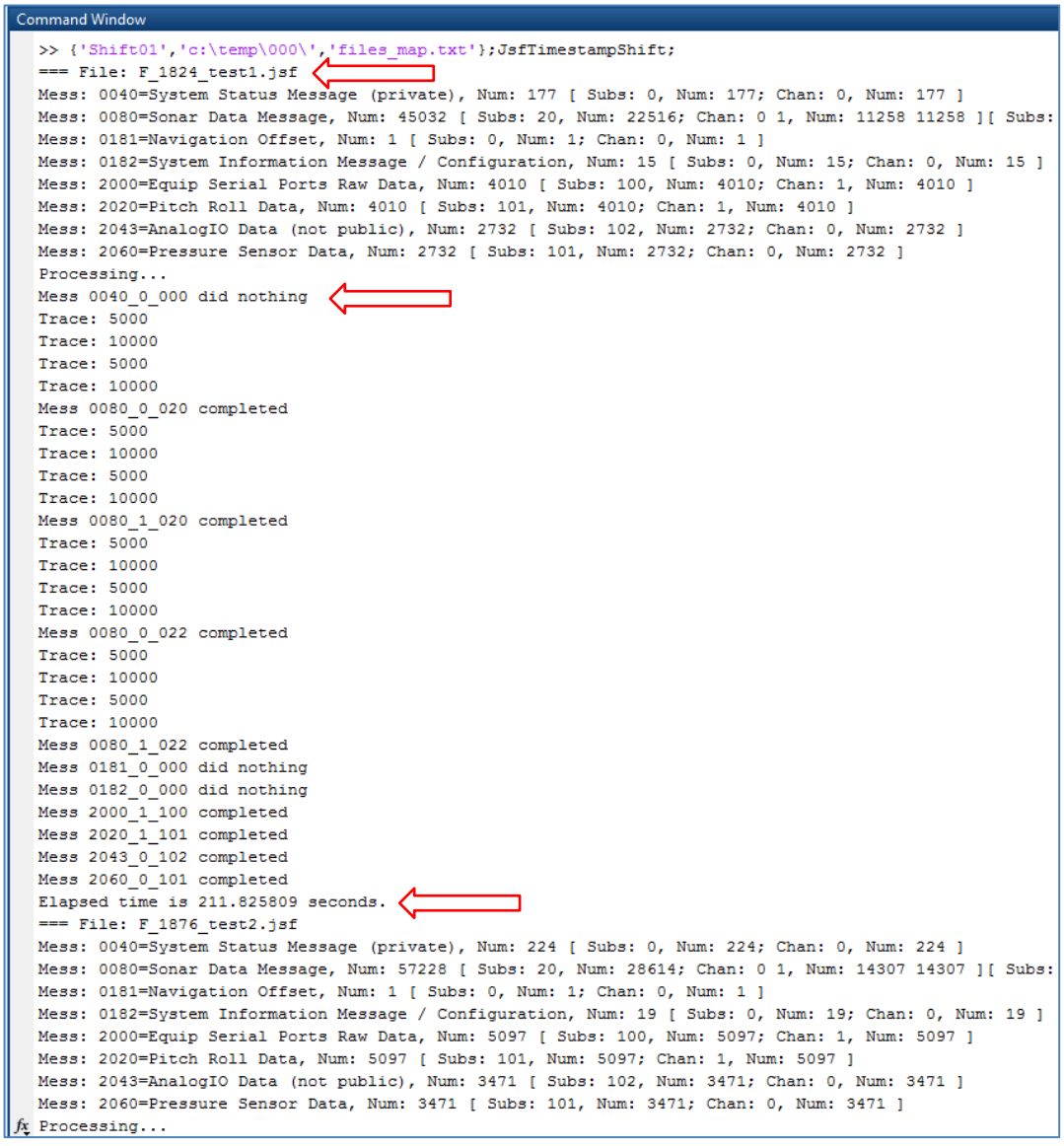


*Figure 1.1* Script execution log

**Notes:**

**1)** The information about messages type in a jsf-file can be obtained without using Matlab, e.g., using SonarWiz, menu "Tools–JSF–JSF Packet Info" (*Figure 1.2*). An example of messages type information is shown in *Figure 1.3*.

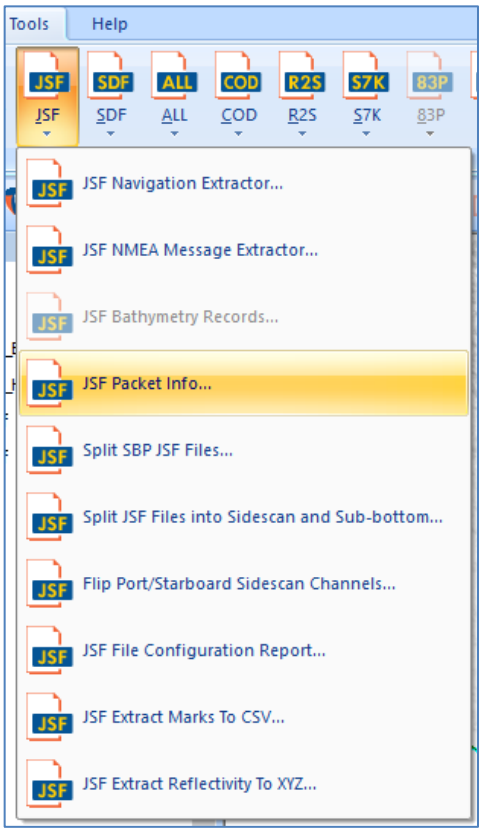


*Figure 1.2* Menu "Tools–JSF–JSF Packet Info"

| | A | B | C |
|---|---|---|---|
| 1 | RecordID | RecordName | Count |
| 2 | 40 | SONAR_MESSAGE_SYSTEM_STATUS | 525 |
| 3 | 80 | SONAR_MESSAGE_DATA | 134376 |
| 4 | 181 | Unknown Message | 1 |
| 5 | 182 | SYSTEM_MESSAGE_SONAR_CONFIGURATION | 41 |
| 6 | 2000 | DEV_SERIAL_DATA | 11966 |
| 7 | 2002 | DEV_SERIAL_NMEA_DATA | 1197 |
| 8 | 2020 | DEV_PITCHROLL_DATA | 11966 |
| 9 | 2043 | DEV_ANALOGIO_DATA_V2 | 8151 |
| 10 | 2060 | DEV_PRESSURE_DATA | 8151 |
| 11 | | | |

*Figure 1.3* The description of messages for ET4205 (created by SonarWiz)

**2)** In SonarWiz it is possible to perform "timestamps shift" for coordinates using NavIvjector by entering an offset in the "Offset in seconds" field (*Figure 1.4*). This will shift the "coordinates time" to the timestamps in the jsf-file (without the timestamps correction within the file). The shift value is input in integer seconds.

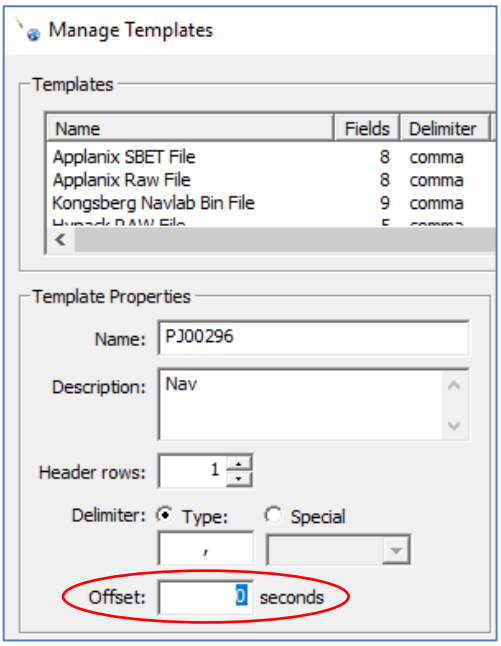


*Figure 1.4* Template settings for CNV-files in NavIvjector

**3)** Sometimes, when working with SonarWiz, CNV files that contain time, coordinates and SSS heading are used (before loading jsf-files into SonarWiz, data from CNV-files is added to them using NavInjector). A common mistake in this case is to write the time (which is output by QINSY or other navigation system) in the HH:MM:SS format without decimal digits for seconds. In this case, the fractional part is truncated, and instead of the "00:00:40.25" entry the CNV-file contains "00:00:40". In this case, the time in the CNV-file is shifted relatively to the UTC time. Theoretically, such a shift can be as large as 1 second (which will give an error of 2.5 meters for a SSS towing speed of 5 knots); in practice, the shift is usually less than 0.1 second (depending on the QINSy configuration). Problems start if the time changes in the area of 00.00 seconds. For example, the sequence of seconds [40.01 41.01 41.99 43.01] would be written as [40 41 41 43]. For 41-41 we will get a "stretch zone" on the sonogram (a small number of pings in a time interval), for 41-43 we will get a "compression zone" (a relatively large number of pings in a time interval). An example of an "incorrect" template for NavInjector without decimals for seconds is shown in *Figure 1.5*. An example of compression and stretch zones is shown in *Figure 1.6*.

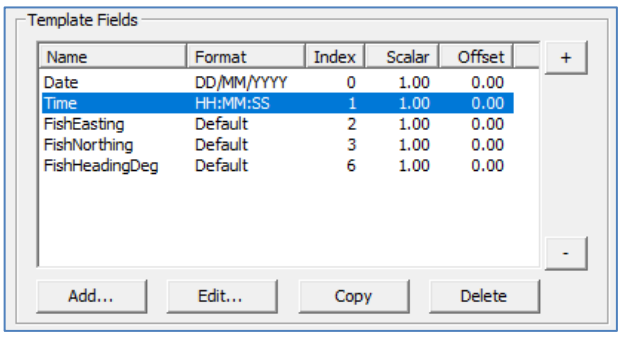


*Figure 1.5* Incorrect template for NavInjector without decimals for seconds

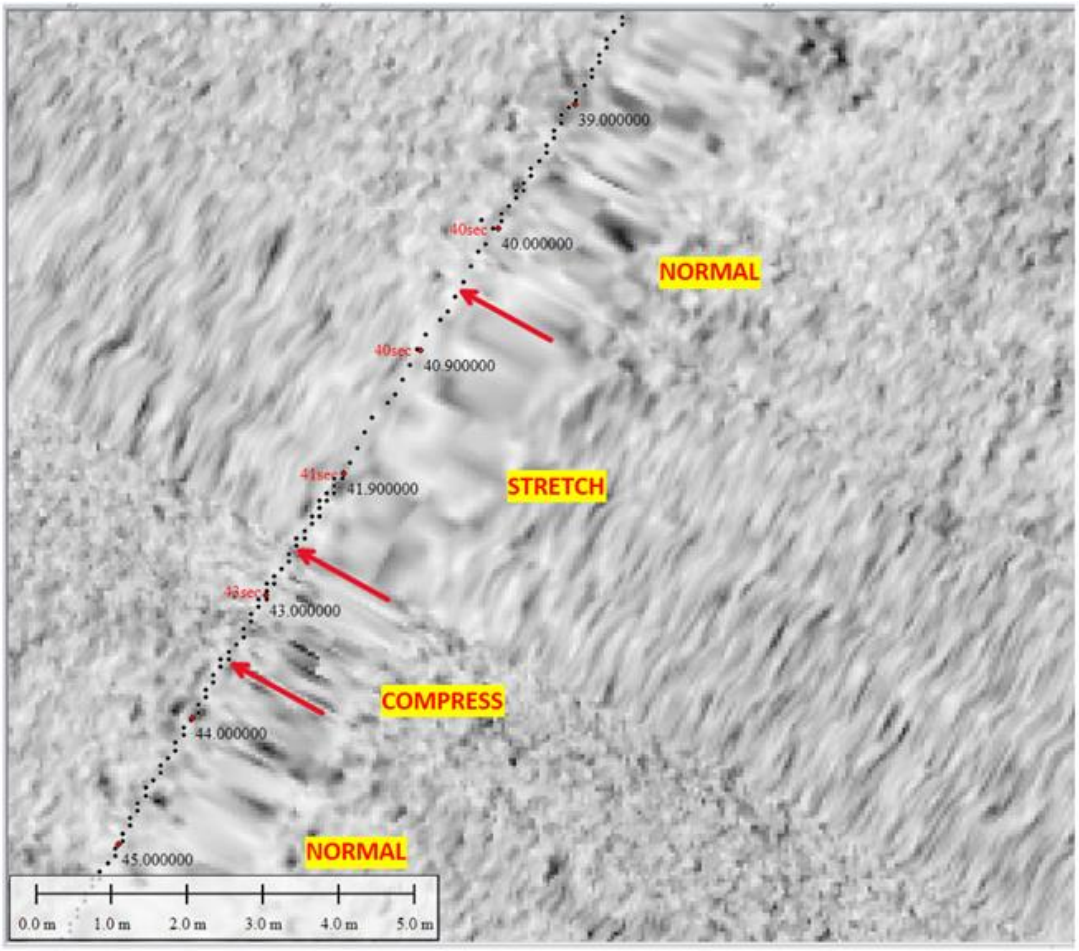


*Figure 1.6* Stretch and compression zones for sonogram due to using time with truncated fractional part of seconds

## 2 Shift compensation technique for timestamps

Let's look at some of the specifics for a time-referencing failure in a jsf-file.

**1)** In the absence of “synchronization” Discoverer software puts the time, determined by the computer clock, inside the jsf-file. Clocks of modern computers are characterized by a linear error accumulation over time (1-2 seconds per day) with small oscillations (probably, oscillations are caused by changes in the temperature of the PCB, supply voltage and other factors). Thus, if the time correction service is not enabled on the computer, then the computer clock deviation in the first approximation can be approximated as a linear function.

**2)** The time shift magnitude can be determined by comparing the SSS mosaic with the MBES data in the nadir region of the sonogram. The position of the seabed features in the nadir zone is dependent weakly on the error presented in the SSS pings rotation angles.

**3)** Time shift compensation by shifting the mosaic in the horizontal plane (correction of sonogram coordinates) can lead to non-uniform “spreading” of targets (SSS and MBES) along the profile, since “nonlinear effects” will be caused by:

-- changes in towing speed along the profile;

-- curvature of the survey line (when performing coordinates correction, for “deviations from a straight line” a parallel translation is performed, and for time correction – “displacement along the track”);

-- SSS course (SSS ping directions) reference offset.

However, in the first approximation, such a shift can give a satisfactory result.

**4)** If the SSS offset is caused not by time errors, but, for example, by incorrect determination of the distance to the SSS by USBL, then compensation of this error by “time offset” can also cause a “spreading” of targets (SSS and MBES) along the profile. In this case, a “spreading” means a change in the offset between the SSS and MBES targets along the profile by a different value, as a result of which the matching of some targets leads to an increase in the discrepancy for other ones.

If jsf-files are found to be “out of sync” with UTC time, then the following steps seem to be optimal for determining the magnitude of the residuals in time and removing them.

**1)** Approximate determination of the time shift value to the nearest second using Offset option in NavInjector (***Figure 1.4***).

**2)** Preparation of a file set using the described script with different shift values within half a second relative to the “approximately determined time-shift value” obtained in the previous step. For example, with [-0.4 -0.2 0 +0.2 +0.4] seconds summands to shift. At this step, we can “inject the shifted time value” only into the message 0080 in the jsf-file (and only into the high-frequency SSS data channel).

**3)** Loading of jsf-files into SonarWiz and selection of the file (time-shift) for which SSS and MBES targets match best.

**4)** Using the time-shift determined in the previous step to inject the shifted time into all recordings of the jsf-file (codes 0080, 2000, 2020, 2043, 2060). Step 4 is presented separately, since for step 2 processing is 3.5 times faster (due to operating with only one message type).

Based on the fact that the computer clock deviates relatively linear, steps 1 or 2 can be performed not for all files, but in 3-6 hours increments. For other files, we can try to perform a linear interpolation of the time-shift values relative to the time the file was recorded. We can control the interpolation result in step 2 or step 4.

When logging the files which are several hours long, a good solution would be to switch from the system “one residual for one file” to the system “the magnitude of residual in time”. Then “shift for timestamp” will be applied not to the whole jsf-file, but for each individual measurement (SSS ping, depth sensor measurement, etc.) inside the jsf-file. In the considered script this approach is not implemented.

# 3 Working with the script

To work with the script, the system of commands of the following format is used:

{CommandName,Parameter1,…,ParameterN};ScriptName;

where

CommandName – script command name;

Parameter1,…,ParameterN – parameters for the script command;

ScriptName – the name of the script.

The script named JsfTimestampShift is used to shift timestamps in a jsf-file. The script can execute three commands:

'Shift01H' – time shift inside jsf file for HF recording (code 0080, subsystem 22, channels 0 and 1);

'Shift01' – time shift inside the jsf file for recordings with codes 0080, 2000, 2020, 2043, 2060;

'InterpShift' – a linear interpolation of the time shift calculated for several files (several timestamps) to all other files (timestamps).

## 3.1 Parameters of commands Shift01 and Shift01H

Example of lines for running the script:

{'Shift01H','c:\temp\000\','files_map.txt'};JsfTimestampShift;

{'Shift01','c:\temp\000\','files_map.txt'};JsfTimestampShift;

where:

-- Shift01H – time shift inside jsf-file for HF recording (code 0080, subsystem 22, channels 0 and 1);

-- Shift01 – time shift inside the jsf-file for recordings with codes 0080, 2000, 2020, 2043, 2060

-- 'c:\temp\000\ – the path to the script root folder, which contains jsf-files for processing and files_map.txt file;

-- files_map.txt – the file in the root folder containing two columns. The first column contains the names of the files being processed (located in the root folder). The second column contains the shift value for the timestamp in seconds to the nearest milliseconds. An example of file is shown in ***Figure 3.1***.

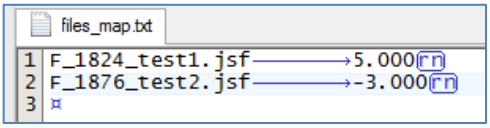


***Figure 3.1*** Example of “files_map.txt” file

Screenshots with an example of processed files are shown in ***Figure 3.2***.

## 3.2 Prosessing of 0040, 0080, 0181, 0182, 2000, 2020, 2043, 2060 messages

The script processes the following types of messages in the jsf-file: 0040, 0080, 0181, 0182, 2000, 2020, 2043, 2060. The types of channels and subsystems containing in the messages are hardcoded in the script. That is, the code to be executed is not universal (it does not independently determine the

types of messages, number of channels and subsystems), but is bound to a specific hardware configuration (when changing the configuration, the script code needs to be modified).

The modified fields/attributes of the messages are described below.

**Mess: 0040=System Status Message (private)**

The message format is not published by EdgeTech. No action is taken.

**Mess: 0080=Sonar Data Message**

A description of the message format is given in "0023492_REV_H, JSF file and message descriptions // EdgeTech, April 27, 2023". The fields/attributes which are modified for the message are shown in ***Table 3.1***.

***Table 3.1*** Modified fields for Message 0080

| Name of field | Description |
|---|---|
| PingTime | Bytes 0-3 // Ping Time in seconds [since the start of time based on time() function] (1/1/1970) (added in protocol version 8, this field is zero in prior protocol versions) |
| Year | Bytes 156-157 // Year Data Recorded (e.g. 2009) (See Bytes 0-3 these 2 time stamps are equivalent and identical). For higher resolution (milliseconds) use the Year, and Day values of bytes 156 to 159, and then use the milliSecondsToday value of bytes 200-203 to complete the timestamp. |
| Day | Bytes 158-159// Day Data Recorded (1–366) (Should not be used) |
| Hour | Bytes 160-161// Hour Data Recorded (see Bytes 200-203) (Should not be used) |
| Minute | Bytes 162-163// Minute Data Recorded (Should not be used) |
| Second | Bytes 164-165// Second Data Recorded (should not be used) |
| MillisecondsToday | Bytes 200-203// Milliseconds today (since midnight) (use in conjunction with Year/Day to get time of Ping) |

**Notes:**

**1)** "PingTime" corresponds to "MillisecondsToday" with a truncated decimal part (similar to the execution of the "fix" function). The fields [Year, Day, Hour, Minute, Second] contain the time corresponding to "PingTime".

**2)** In jsf-files recorded using earlier versions of Discoverer, there was some desynchronization of [Year, Day] with [Hour, Minute, Sec]; in this case, the date change was performed later than 00:00:00. This error has been corrected in subsequent Discoverer versions.

**3)** Probably, the time in the marked fields is measured by computer clock readings, for which the shift is calculated relative to UTC time, obtained from ZDA or GGA navigation messages.

**4)** Since navigation messages are characterized by a delay (which can be as high as 0.1-0.3 seconds), the "shift from UTC time" is probably calculated by averaging over a relatively long period of time and further correction. The smallest delay in sending navigation GGA and GLL messages was characterized by GPS Trimble operating without corrections (less than 0.1 second when operating with corrections); Veripos is characterized by a longer delay (0.1-0.2 sec when operating with corrections); C-Nav is characterized by the greatest delay (up to 0.4 seconds when operating with corrections). In order to

eliminate the impact of such “delays”, it is preferable to use Trimble or another method of registering a PPS signal by the computer clock, followed by time correction inside the jsf-file.

**Mess: 0181=Navigation Offset**

The message format is not published by EdgeTech. The message occurs once at the beginning of the jsf-file and almost always contains a sequence of null values (that is, it does not contain timestamps). No action is taken.

**Mess: 0182=System Information Message**

A partial description of the message format is given in “0023492_REV_H, JSF file and message descriptions // EdgeTech, April 27, 2023”. The message contains information about the equipment configuration (probably mainly about acoustic channels). With a high probability, the message does not contain timestamps. No action is taken.

The document “0023492_REV_H, JSF file and message descriptions // EdgeTech, April 27, 2023” describes the first 6 fields of the message. Subsequent bytes of the message (about a kilobyte) are referred to as “Reserved”. A more complete decoding of the message can be obtained using SonarWiz, menu “Tools–JSF–JSF File Configuration Report”. An example of decryption is given in Appendix 1.

**Mess: 2000=Equip Serial Ports Raw Data**

The message format is not published by EdgeTech. It has been empirically established that the message contains data received by the towed device from the com-port (for example, a magnetometer connected to the piggy-back). The format is fully consistent with the message 2020 described in “0023492_REV_H, JSF file and message descriptions // EdgeTech, April 27, 2023”. The fields/attributes which are modified for the message are shown in ***Table 3.2***.

For ET4205 configuration considered, message 2000 contains the TSS1 format string with the readings of the internal compass-tiltmeter. The message 2000 always follows 2020.

***Table 3.2*** Modified fields for Messages 20xx

| Name of field | Description |
|---|---|
| TimeInSeconds | Bytes 0-3 // Ping Time in seconds [since the start of time based on time() function] (1/1/1970)) |
| MillisecondsCurrentSecond | Bytes 4-7 // Milliseconds in the current second |

**Mess: 2020=Pitch Roll Data**

A description of the message format is given in “0023492_REV_H, JSF file and message descriptions // EdgeTech, April 27, 2023”. The fields/attributes modified for the message are shown in ***Table 3.2***.

**Mess: 2043=AnalogIO Data**

The message format is not published by EdgeTech. SonarWiz output the name “AnalogIO Data” for the message (***Figure 1.3***). The number of messages in the file is equal to the number of messages 2060, and they also follow the messages 2060 inside the file. By analogy with messages 2020 & 2000, it

can be assumed that 2043 contains raw data from the pressure sensor, which in “decrypted form” is contained in 2060. The timestamp was detected in bytes given in ***Table 3.2***; timestamps are corrected for these bytes.

**Mess: 2060=Pressure Sensor Data**

A description of the message format is given in “0023492_REV_H, JSF file and message descriptions // EdgeTech, April 27, 2023”. The fields/attributes which are modified for the message are shown in ***Table 3.2***.

## 3.3 Results of the Shift01 and Shift01H commands execution

As a result of the script execution, jsf-files are created in the root folder (specified in the parameters when the script was run) with the entered timestamp shift (according to the shift values specified in the “files_map.txt” file).

For the Shift01H command, a postfix is added to the name of the processed file with the timestamp shift value in seconds. An example of a source file name is “F_1824_test1.jsf”, an example of a file name after a timestamp shift is “F_1824_test1_+5.200.jsf” (shift +5.200 seconds). ***Figure 3.2*** shows two screenshots. The first screenshot (on the left side) is the view of the root folder before running the script. The second screenshot is the view of the root folder after running the script; the folder contains temporary files (*.tmp extension) and processed files. Two different timestamp shifts were applied to each source file (***Figure 3.3***).

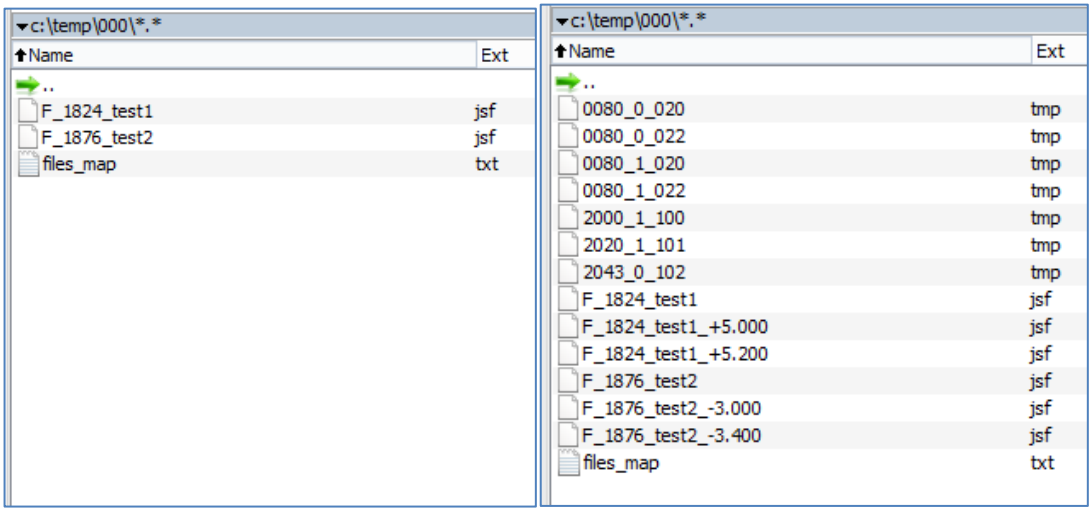


***Figure 3.2*** The root folder initial view (input data only, left side) and root folder view with files processed (right side)

For the Shift01 command, the postfix “_shift” is added to the name of the processed file. That is, it is assumed that one time-shift will be applied to one file name. The time in the file attributes is shifted as well.

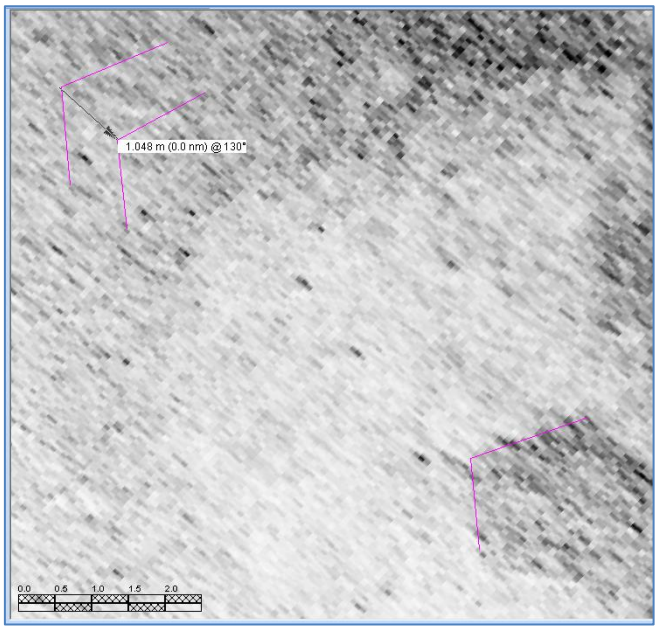


*Figure 3.3* The seabed feature marked for the source jsf-file and for files with the entered time shift of -3.0 and -3.4 seconds

## 3.4 Parameters of InterpShift command

The command performs a linear interpolation of the time-shift calculated for several files (several timestamps) to all other files (timestamps).

Example of lines for running the script:

{'InterpShift','c:\temp\000\interp.txt'};JsfTimestampShift;

where

-- InterpShift – interpolation command name;

-- 'c:\temp\000\interp.txt' – the path to a text file containing four columns: date, time, file name, shift value for the timestamp in seconds to the nearest milliseconds (or the 'nan' value, if the shift value is undefined). An example of the file is shown in *Figure 3.1*.

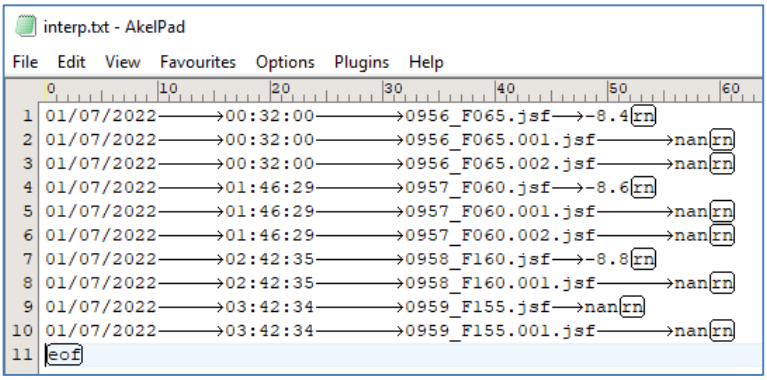


*Figure 3.4* Example of the “interp.txt” file

The column with the interpolation result is written to a file with the “*.txt01” extension. The result can be copied to Excel and used for the files_map.txt file preparation and subsequent Shift01 command execution.

## 4 Method of high-precision time referencing of sonograms

Three "time sources" can be used for time referencing while surveying using ET Discoverer: (1) computer clock, (2) timestamp transmitted in the ZDA string, (3) timestamp transmitted with SSS coordinates in the GGA or GLL type string.

The algorithm of interaction between the three time sources is not described by the manufacturer. We can assume that the **following rules for time synchronization** are fulfilled:

**1)** If there are no ZDA, GGA, GLL messages, the computer clock time is used;

**2)** When a ZDA message is received, the computer clock is periodically synchronized with the time received in the ZDA message. The frequency of this synchronization is undefined; it is also undefined whether synchronization is performed on data in a single ZDA message, or whether statistics are collected from ZDA message data over a certain period of time. It seems reasonable to create an algorithm in which synchronization is related to the Windows Time service. In this case, synchronization is performed according to the following rules (https://learn.microsoft.com/en-us/windows-server/networking/windows-time-service/windows-time-service-tools-and-settings?tabs=config):

-- the CurrentTimeOffset offset is checked (current time and computer time) to determine if the offset is less than the MaxAllowedPhaseOffset threshold;

-- if CurrentTimeOffset<MaxAllowedPhaseOffset, then the computer clock is gradually adjusted using the "clock speed" (the value of PhaseCorrection, the algorithm differs from one Windows version to another);

-- if CurrentTimeOffset>MaxAllowedPhaseOffset, then the computer clock is set immediately.

In this case, the computer clock will be corrected with a significant deviation by the first ZDA message (indeed, Discoverer performs an “immediate correction” with an error of a few seconds). By subsequent ZDA messages, a "smooth adjustment" of the computer clock will be performed by the PhaseCorrection value correction (indeed, when a ZDA is received with an error of 1 second, the computer clock is not adjusted - the time recorded in the 0080 message body differs from the time received in the "ZDA message with error").

**3)** When time is received in GGA or GLL, it is ignored if ZDA is received. If the ZDA is not received, it is used to adjust the computer clock as described above.

Possible **sources of time referencing error**:

**1)** Time referencing is performed by the ZDA message receiving from the navigation system (not from the GPS receiver). For example, when a message is received from QINSy, there are the effects of “time repetition” (when two consecutive ZDA messages for different seconds contain the same time) and “time slip” (time in the ZDA message is slowly “shifted” with subsequent instantaneous correction). The described effects occur irregularly, however, they will contribute to "clock adjustment" using PhaseCorrection. Most likely, it will be some "statistical lag" of the computer clock from the exact time.

**2)** Receiving delay of ZDA, GGA, GLL messages from the navigation receiver. We did not collect the experimental statistics for the ZDA message. For navigation messages GGA and GLL, the lowest delay

was characterized by GPS Trimble, it was less than 0.1 second when operating with corrections; Veripos is characterized by a delay of 0.1-0.2 seconds when operating with corrections; C-Nav is characterized by the greatest delay - up to 0.4 seconds when operating with corrections. That is, when receiving time from GGA or GLL messages with C-Nav (when operating with corrections), the average delay can be approximately 0.3 seconds (approximately 0.6 meters at 4 knots).

Thus, when receiving a ZDA message directly from Trimble, the sonogram ping error will be less than 0.1 seconds (0.2 meters at a towing speed of 4 knots), which is sufficient for most types of engineering surveys with a towed SSS.

Since the Discoverer program is usually installed on a different computer than the navigation system, there may be cases when highly accurate time referencing is required. For example, if we are surveying with a ROV, then a computer clock synchronization error will affect the MRU data referencing and Heading-Pitch-Roll angle referencing accuracy. Considering the “fast corrective turns” performed by the ROV autopilot (for example, due to the impact of a side current), a time referencing error of 0.1-0.3 seconds can be critical for determining the angle of the SSS pings direction. For example, theoretically, an angle error of 2 degrees due to the time lag would result in a displacement of 0.9 meters at the edges of a sonogram with a slant range of 25 meters. In practice, the desynchronization of SSS and MRU time leads to the “doubling” of targets on sonograms due to the yaw of the ROV, as well as to the “stretching” of local objects and incorrect determination of their sizes.

**To perform high-precision synchronization, the following method can be used:**

-- SSS data recording is performed using Discoverer without navigation messages;
-- the Windows time service is disabled;
-- the ZDA message is logged using the gLog program (the ZDA messages and the receiving time of the message’s first byte are recorded according to the computer clock);
-- PPS logging is performed using the FixBox (byte with 0 value, received by the computer's COM port) and the gLog program;
-- time stamps processing is performed: ZDA receiving stability is estimated according to the computer clock (absence of high-frequency fluctuations); the ZDA time is referenced to the moments of PPS reception; the computer clock is referenced to the ZDA-PPS time; the exact time is being injected into the Jsf-file.

## Appendix 1. Content of the 0182 message (created by SonarWiz)

JSF Configuration Report: D:\temp.jsf
System Type: (51): Unknown system type
Low Rate IO Enabled: 1
Software Version: 795
Number of SubSystems: 3
Number of Serial Ports: 2
System Serial Number: 50009
Flags and Options: 1

Detail for 3 SubSystem(s)...
1. SubSystem Number: 20
    Sonar On: 1
    NetworkWindowInfo (Channel 0 only)...
        Frame Decimation: 1
        Pixel Decimation: 1
        Skip Samples: 0
        Total Samples: 2136
        Max Packet Size: 4272
    Network Window Enables: 1
    Disk Window Info (Channel 0 only)...
        Frame Decimation: 0
        Pixel Decimation: 0
        Skip Samples: 0
        Total Samples: 0
        Max Packet Size: 0
    Disk Window Enables: 16
    Post-processor Mode: 0
    Data Type CH0: 0 [one short per sample - envelope data. No ADC required]
    Bit Flags: 30
    Ping Rate or Range: 40000 (kHz or mm)
        Coupling Parameters
        Subsystem number to trigger this subsystem on: 20
        Trigger on every Nth: 1
        Trigger delay (uS): 0
    Trigger Mode: 0 (0 for internal, 1 for external, 2 for coupled)
    Direct Path Removal: -2
    Direct Path Value: 0
    Transmit Gains: [1000,1000,0,0]
    Total Transmit Gains: [449,449,0,0]
    ADC Gains: [32000,32000,32000,32000]
        Pulse Info
        Pulse File: 4205_SS230Q__25BW__1_5MS.spf
        Pulse Type: 18
        Number of Pulses: 1
        Max Ping Rate: 20.00 (Hz)
        Data Rate: 40.06 (Hz)
        Min Frequency: [217500.00,217500.00] (Hz)
        Max Frequency: [242500.00,242500.00] (Hz)
        Pulse Duration: [1.50,1.50] (ms)
        Pulse ID: 50860 (Hz)
        Pulse Desc: SS230Q 230.0 kHz:BW 25.0 kHz: 1.5 ms:ID 50860:SR 40.1:OL 100%
        System Type Pulse Designed For: 76
        Option Bits: 0

2. SubSystem Number: 21
    Sonar On: 0
    NetworkWindowInfo (Channel 0 only)...
        Frame Decimation: 1

Pixel Decimation: 1
Skip Samples: 0
Total Samples: 4166
Max Packet Size: 8332
Network Window Enables: 1
Disk Window Info (Channel 0 only)...
Frame Decimation: 0
Pixel Decimation: 0
Skip Samples: 0
Total Samples: 0
Max Packet Size: 0
Disk Window Enables: 16
Post-processor Mode: 0
Data Type CH0: 0 [one short per sample - envelope data. No ADC required]
Bit Flags: 10
Ping Rate or Range: 40000 (kHz or mm)
Coupling Parameters
Subsystem number to trigger this subsystem on: 20
Trigger on every Nth: 1
Trigger delay (uS): 0
Trigger Mode: 2 (0 for internal, 1 for external, 2 for coupled)
Direct Path Removal: -2
Direct Path Value: 0
Transmit Gains: [1000,1000,0,0]
Total Transmit Gains: [600,600,0,0]
ADC Gains: [32000,32000,32000,32000]
Pulse Info
Pulse File: 4205_SS540Q__60BW__0_5MS.spf
Pulse Type: 18
Number of Pulses: 1
Max Ping Rate: 60.00 (Hz)
Data Rate: 78.13 (Hz)
Min Frequency: [510000.00,510000.00] (Hz)
Max Frequency: [570000.00,570000.00] (Hz)
Pulse Duration: [0.50,0.50] (ms)
Pulse ID: 50865 (Hz)
Pulse Desc: SS540Q 540.0 kHz:BW 60.0 kHz: 0.5 ms:ID 50865:SR 78.1:OL 100%
System Type Pulse Designed For: 73
Option Bits: 0

3. SubSystem Number: 22
Sonar On: 1
NetworkWindowInfo (Channel 0 only)...
Frame Decimation: 1
Pixel Decimation: 1
Skip Samples: 0
Total Samples: 4902
Max Packet Size: 9804
Network Window Enables: 1
Disk Window Info (Channel 0 only)...
Frame Decimation: 0
Pixel Decimation: 0
Skip Samples: 0
Total Samples: 0
Max Packet Size: 0
Disk Window Enables: 16
Post-processor Mode: 0
Data Type CH0: 0 [one short per sample - envelope data. No ADC required]
Bit Flags: 30
Ping Rate or Range: 40000 (kHz or mm)
Coupling Parameters

Subsystem number to trigger this subsystem on: 20
Trigger on every Nth: 1
Trigger delay (uS): 0
Trigger Mode: 2 (0 for internal, 1 for external, 2 for coupled)
Direct Path Removal: -2
Direct Path Value: 0
Transmit Gains: [1000,1000,0,0]
Total Transmit Gains: [540,540,0,0]
ADC Gains: [32000,32000,32000,32000]
Pulse Info
Pulse File: 4205_SS850__75BW__1_60MS.spf
Pulse Type: 18
Number of Pulses: 1
Max Ping Rate: 18.75 (Hz)
Data Rate: 91.91 (Hz)
Min Frequency: [812500.00,812500.00] (Hz)
Max Frequency: [887500.00,887500.00] (Hz)
Pulse Duration: [1.60,1.60] (ms)
Pulse ID: 50875 (Hz)
Pulse Desc: SS: 850KHz, Bandwidth:75KHz, Length:1.6ms, ID:50875
System Type Pulse Designed For: 75
Option Bits: 0

Detail for 2 Serial Port(s)...
1. Logical Channel: 1
Port Number: 4
Baud Rate: 19200
Parser Algorithm: (4): P
Port Flags: 3B
Port Type: 1
IP4: 0.0.0.0
Latency: 0.000 (s)

2. Logical Channel: 5
Port Number: 3
Baud Rate: 9600
Parser Algorithm: (0): N
Port Flags: 3B
Port Type: 1
IP4: 0.0.0.0
Latency: 0.000 (s)

## Appendix 2. Script code

```
%script JsfTimestampShift;
%The script is solving follow task: shift timestamp in a jsf-files. There are follow messages in file:
%0040=System Status Message (private), 0080=Sonar Data Message, 0181=Navigation Offset, 0182=System Information Message / Configuration,
%2000=Equip Serial Ports Raw Data, 2020=Pitch Roll Data, 2043=AnalogIO Data (not public), 2060=Pressure Sensor Data.
%Commands example:
%{'Shift01H','c:\temp\000\','files_map.txt'};JsfTimestampShift;
%{'InterpShift','c:\temp\000\interp.txt'};JsfTimestampShift;
%{'Shift01','c:\temp\000\','files_map.txt'};JsfTimestampShift;
%Input parameters for 'Shift01H' and 'Shift01' command:
%  c:\temp\000\ - the name of folder with jsf-files;
%  files_map.txt - ASCII file included two columns: file name and shift with 3 digits (for example 5.230), delimiter TAB.

gKey=ans;
if strcmp(gKey{1},'Shift01') %shift timestamp in a jsf-files
    %{'Shift01','c:\temp\000\','files_map.txt'};JsfTimestampShift;
    try dName=gKey{2};catch,dName=input('The name of folder with jsf-files=');end; try fNameMap=gKey{3};catch,fNameMap=input('The ASCII with jsf-file
names and shift=');end;
    fId=fopen([dName
fNameMap],'r');C=textscan(fId,'%s%f','Delimiter','\t','MultipleDelimsAsOne',0,'EndOfLine','\r\n');fclose(fId);fName=C{1};LL=numel(fName);ddSec=C{2};
    for nn=1:LL,
        tic;dSec=round(ddSec(nn).*1000)./1000;disp(['=== File: ' fName{nn} num2str(dSec,'%+9.3f')]);gJsfHeaderRead([dName
fName{nn}],1);disp('Processing...');
        %Mess: 0040=System Status Message (private), Num: 304 [ Subs: 0, Num: 304; Chan: 0, Num: 304 ]
        disp('Mess 0040_0_000 did nothing');
        %=== Message 0080 timestamp bytes:
        %Head.PingTime(m)=fread(fId,1,'int32'); %0-3// Ping Time in seconds [since the start of time based on time() function] (1/1/1970) (added in
protocol version 8,  this  field  is  zero  in  prior  protocol versions)
        %Head.Year(m)=fread(fId,1,'int16'); %156-157// Year Data Recorded (e.g. 2009) (See Bytes 0-3 these 2 time stamps are equivalent and identical).
For higher resolution (milliseconds) use the Year, and Day values of bytes 156 to 159, and then use the milliSecondsToday value of bytes 200-203 to
complete the timestamp.
        %Head.Day(m)=fread(fId,1,'int16'); %158-159// Day Data Recorded (1-366) (Should not be used)
        %Head.Hour(m)=fread(fId,1,'int16'); %160-161// Hour Data Recorded (see Bytes 200-203) (Should not be used)
        %Head.Minute(m)=fread(fId,1,'int16'); %162-163// Minute Data Recorded (Should not be used)
        %Head.Second(m)=fread(fId,1,'int16'); %164-165// Second Data Recorded (should not be used)
        %Head.MillisecondsToday(m)=fread(fId,1,'uint32'); %200-203// Milliseconds today (since midnight) (use in conjunction with Year/Day to get time
of Ping)
        %Mess: 0080=Sonar Data Message, Num: 77666 [ Subs: 20, Num: 38832; Chan: 0 1, Num: 19416 19416 ][ Subs: 22, Num: 38834; Chan: 0 1, Num: 19417
19417 ]
        JsfHead=gJsfHeaderRead([dName fName{nn}],0);[Head,Data]=gJsf0080Read(JsfHead,0,20);
        [~,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);if ~all(Sd==fix(Head.MillisecondsToday./1000)),error('Incorrect INITIAL time for
0080_0_020');end;
        tmp1=mod(Head.MillisecondsToday,1000)+round(dSec.*1000);tmp2=mod(tmp1,1000);MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.PingTime=Head.PingTime+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation for
0080_0_020');end;
        [Dx,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);[Head.Year,~,~]=datevec(Dx);Head.Day=Dx-
datenum(Head.Year,ones(size(Head.Year)),ones(size(Head.Year)))+1;[Head.Hour,Head.Minute,Head.Second]=gNavTime2Time('Sd2HMS3',Sd);
```

```
        Head.MillisecondsToday=Head.Hour.*3600000+Head.Minute.*60000+Head.Second.*1000+MillisecondsCurrentSecond;
        gJsf0080Write(JsfHead,Head,Data,[dName '0080_0_020.tmp'],0);disp('Mess 0080_0_020 completed');
        %Mess: 0080=Sonar Data Message, Num: 77666 [ Subs: 20, Num: 38832; Chan: 0 1, Num: 19416 19416 ][ Subs: 22, Num: 38834; Chan: 0 1, Num: 19417
19417 ]
        JsfHead=gJsfHeaderRead([dName '0080_0_020.tmp'],0);[Head,Data]=gJsf0080Read(JsfHead,1,20);
        [~,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);if ~all(Sd==fix(Head.MillisecondsToday./1000)),error('Incorrect INITIAL time for
0080_1_020');end;
        tmp1=mod(Head.MillisecondsToday,1000)+round(dSec.*1000);tmp2=mod(tmp1,1000);MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.PingTime=Head.PingTime+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation for
0080_1_020');end;
        [Dx,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);[Head.Year,~,~]=datevec(Dx);Head.Day=Dx-
datenum(Head.Year,ones(size(Head.Year)),ones(size(Head.Year)))+1;[Head.Hour,Head.Minute,Head.Second]=gNavTime2Time('Sd2HMS3',Sd);
        Head.MillisecondsToday=Head.Hour.*3600000+Head.Minute.*60000+Head.Second.*1000+MillisecondsCurrentSecond;
        gJsf0080Write(JsfHead,Head,Data,[dName '0080_1_020.tmp'],0);disp('Mess 0080_1_020 completed');
        %Mess: 0080=Sonar Data Message, Num: 77666 [ Subs: 20, Num: 38832; Chan: 0 1, Num: 19416 19416 ][ Subs: 22, Num: 38834; Chan: 0 1, Num: 19417
19417 ]
        JsfHead=gJsfHeaderRead([dName '0080_1_020.tmp'],0);[Head,Data]=gJsf0080Read(JsfHead,0,22);
        [~,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);if ~all(Sd==fix(Head.MillisecondsToday./1000)),error('Incorrect INITIAL time for
0080_0_022');end;
        tmp1=mod(Head.MillisecondsToday,1000)+round(dSec.*1000);tmp2=mod(tmp1,1000);MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.PingTime=Head.PingTime+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation for
0080_0_022');end;
        [Dx,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);[Head.Year,~,~]=datevec(Dx);Head.Day=Dx-
datenum(Head.Year,ones(size(Head.Year)),ones(size(Head.Year)))+1;[Head.Hour,Head.Minute,Head.Second]=gNavTime2Time('Sd2HMS3',Sd);
        Head.MillisecondsToday=Head.Hour.*3600000+Head.Minute.*60000+Head.Second.*1000+MillisecondsCurrentSecond;
        gJsf0080Write(JsfHead,Head,Data,[dName '0080_0_022.tmp'],0);disp('Mess 0080_0_022 completed');
        %Mess: 0080=Sonar Data Message, Num: 77666 [ Subs: 20, Num: 38832; Chan: 0 1, Num: 19416 19416 ][ Subs: 22, Num: 38834; Chan: 0 1, Num: 19417
19417 ]
        JsfHead=gJsfHeaderRead([dName '0080_0_022.tmp'],0);[Head,Data]=gJsf0080Read(JsfHead,1,22);
        [~,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);if ~all(Sd==fix(Head.MillisecondsToday./1000)),error('Incorrect INITIAL time for
0080_1_022');end;
        tmp1=mod(Head.MillisecondsToday,1000)+round(dSec.*1000);tmp2=mod(tmp1,1000);MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.PingTime=Head.PingTime+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation for
0080_1_022');end;
        [Dx,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);[Head.Year,~,~]=datevec(Dx);Head.Day=Dx-
datenum(Head.Year,ones(size(Head.Year)),ones(size(Head.Year)))+1;[Head.Hour,Head.Minute,Head.Second]=gNavTime2Time('Sd2HMS3',Sd);
        Head.MillisecondsToday=Head.Hour.*3600000+Head.Minute.*60000+Head.Second.*1000+MillisecondsCurrentSecond;
        gJsf0080Write(JsfHead,Head,Data,[dName '0080_1_022.tmp'],0);disp('Mess 0080_1_022 completed');
        %Mess: 0181=Navigation Offset, Num: 1 [ Subs: 0, Num: 1; Chan: 0, Num: 1 ]
        disp('Mess 0181_0_000 did nothing');
        %Mess: 0182=System Information Message / Configuration, Num: 23 [ Subs: 0, Num: 23; Chan: 0, Num: 23 ]
        disp('Mess 0182_0_000 did nothing');
        %=== Messages 20xx timestamp bytes:
        %Head.TimeInSeconds(m)=fread(fId,1,'int32'); %0-3// Time in seconds (since the start of time based on time() function) (1/1/1970)
        %Head.MillisecondsCurrentSecond(m)=fread(fId,1,'int32'); %4-7// Milliseconds in the current second
        %Mess: 2000=Equip Serial Ports Raw Data, Num: 6917 [ Subs: 100, Num: 6917; Chan: 1, Num: 6917 ]
        JsfHead=gJsfHeaderRead([dName '0080_1_022.tmp'],0);Head=gJsf20xxRead(JsfHead,2000,1,100);
        tmp1=Head.MillisecondsCurrentSecond+round(dSec.*1000);tmp2=mod(tmp1,1000);Head.MillisecondsCurrentSecond=tmp2;
```

```
        tmp3=(tmp1-tmp2)./1000;Head.TimeInSeconds=Head.TimeInSeconds+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation
for 2000_1_100');end;
        gJsf20xxWrite(JsfHead,Head,[dName '2000_1_100.tmp'],0);disp('Mess 2000_1_100 completed');
        %Mess: 2020=Pitch Roll Data, Num: 6917 [ Subs: 101, Num: 6917; Chan: 1, Num: 6917 ]
        JsfHead=gJsfHeaderRead([dName '2000_1_100.tmp'],0);Head=gJsf20xxRead(JsfHead,2020,1,101);
        tmp1=Head.MillisecondsCurrentSecond+round(dSec.*1000);tmp2=mod(tmp1,1000);Head.MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.TimeInSeconds=Head.TimeInSeconds+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation
for 2020_1_101');end;
        gJsf20xxWrite(JsfHead,Head,[dName '2020_1_101.tmp'],0);disp('Mess 2020_1_101 completed');
        %Mess: 2043=AnalogIO Data (not public), Num: 4711 [ Subs: 102, Num: 4711; Chan: 0, Num: 4711 ]
        JsfHead=gJsfHeaderRead([dName '2020_1_101.tmp'],0);Head=gJsf20xxRead(JsfHead,2043,0,102);
        tmp1=Head.MillisecondsCurrentSecond+round(dSec.*1000);tmp2=mod(tmp1,1000);Head.MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.TimeInSeconds=Head.TimeInSeconds+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation
for 2043_0_102');end;
        gJsf20xxWrite(JsfHead,Head,[dName '2043_0_102.tmp'],0);disp('Mess 2043_0_102 completed');
        %Mess: 2060=Pressure Sensor Data, Num: 4711 [ Subs: 101, Num: 4711; Chan: 0, Num: 4711 ]
        JsfHead=gJsfHeaderRead([dName '2043_0_102.tmp'],0);Head=gJsf20xxRead(JsfHead,2060,0,101);
        tmp1=Head.MillisecondsCurrentSecond+round(dSec.*1000);tmp2=mod(tmp1,1000);Head.MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.TimeInSeconds=Head.TimeInSeconds+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation
for 2060_0_101');end;
        gJsf20xxWrite(JsfHead,Head,[dName fName{nn}(1:end-4) '_shift.jsf'],0);disp('Mess 2060_0_101 completed'); %'_' num2str(dSec,'%+9.3f')
        %Time for file in powershell (Windows) // to get time in 100uSec use !powershell (Get-Item
c:\temp\000\01_2060_0_101.tmp).CreationTime.ToFileTime()
        system(['powershell (Get-Item ' dName fName{nn}(1:end-4) '_shift.jsf).CreationTime=(Get-Item ' dName fName{nn} ').CreationTime.AddSeconds('
num2str(fix(dSec),'%9.0f') ').AddMilliseconds(' num2str(rem(dSec.*1000,1000),'%9.0f') ')']); %'_' num2str(dSec,'%+9.3f')
        system(['powershell (Get-Item ' dName fName{nn}(1:end-4) '_shift.jsf).LastWriteTime=(Get-Item ' dName fName{nn} ').LastWriteTime.AddSeconds('
num2str(fix(dSec),'%9.0f') ').AddMilliseconds(' num2str(rem(dSec.*1000,1000),'%9.0f') ')']);
        system(['powershell (Get-Item ' dName fName{nn}(1:end-4) '_shift.jsf).LastAccessTime=(Get-Item ' dName fName{nn} ').LastAccessTime.AddSeconds('
num2str(fix(dSec),'%9.0f') ').AddMilliseconds(' num2str(rem(dSec.*1000,1000),'%9.0f') ')']);
        toc;
    end;
    clearvars dName fNameMap fId C fName LL ddSec nn dSec JsfHead Head Data Sd tmp1 tmp2 MillisecondsCurrentSecond tmp3 Dx Sd
end;
if strcmp(gKey{1},'Shift01H') %shift timestamp in a jsf-files for HF only
    %{'Shift01H','c:\temp\000\','files_map.txt'};JsfTimestampShift;
    try dName=gKey{2};catch,dName=input('The name of folder with jsf-files=');end; try fNameMap=gKey{3};catch,fNameMap=input('The ASCII with jsf-file
names and shift=');end;
    fId=fopen([dName
fNameMap],'r');C=textscan(fId,'%s%f','Delimiter','\t','MultipleDelimsAsOne',0,'EndOfLine','\r\n');fclose(fId);fName=C{1};LL=numel(fName);ddSec=C{2};
    for nn=1:LL,
        tic;dSec=round(ddSec(nn).*1000)./1000;disp(['=== File: ' fName{nn} num2str(dSec,'%+9.3f')]);disp('Processing...');
        %Mess: 0080=Sonar Data Message, Num: 77666 [ Subs: 20, Num: 38832; Chan: 0 1, Num: 19416 19416 ][ Subs: 22, Num: 38834; Chan: 0 1, Num: 19417
19417 ]
        JsfHead=gJsfHeaderRead([dName fName{nn}],1);[Head,Data]=gJsf0080Read(JsfHead,0,22);
        [~,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);if ~all(Sd==fix(Head.MillisecondsToday./1000)),error('Incorrect INITIAL time for
0080_0_022');end;
        tmp1=mod(Head.MillisecondsToday,1000)+round(dSec.*1000);tmp2=mod(tmp1,1000);MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.PingTime=Head.PingTime+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation for
0080_0_022');end;
```

```
        [Dx,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);[Head.Year,~,~]=datevec(Dx);Head.Day=Dx-
datenum(Head.Year,ones(size(Head.Year)),ones(size(Head.Year)))+1;[Head.Hour,Head.Minute,Head.Second]=gNavTime2Time('Sd2HMS3',Sd);
        Head.MillisecondsToday=Head.Hour.*3600000+Head.Minute.*60000+Head.Second.*1000+MillisecondsCurrentSecond;
        gJsf0080Write(JsfHead,Head,Data,[dName '0080_0_022.tmp'],0);disp('Mess 0080_0_022 completed');
        %Mess: 0080=Sonar Data Message, Num: 77666 [ Subs: 20, Num: 38832; Chan: 0 1, Num: 19416 19416 ][ Subs: 22, Num: 38834; Chan: 0 1, Num: 19417
19417 ]
        JsfHead=gJsfHeaderRead([dName '0080_0_022.tmp'],0);[Head,Data]=gJsf0080Read(JsfHead,1,22);
        [~,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);if ~all(Sd==fix(Head.MillisecondsToday./1000)),error('Incorrect INITIAL time for
0080_1_022');end;
        tmp1=mod(Head.MillisecondsToday,1000)+round(dSec.*1000);tmp2=mod(tmp1,1000);MillisecondsCurrentSecond=tmp2;
        tmp3=(tmp1-tmp2)./1000;Head.PingTime=Head.PingTime+round(tmp3);if ~all(abs(tmp3-round(tmp3))<0.001),error('Incorrect time calculation for
0080_1_022');end;
        [Dx,Sd]=gNavTime2Time('Ut2DxSd',Head.PingTime);[Head.Year,~,~]=datevec(Dx);Head.Day=Dx-
datenum(Head.Year,ones(size(Head.Year)),ones(size(Head.Year)))+1;[Head.Hour,Head.Minute,Head.Second]=gNavTime2Time('Sd2HMS3',Sd);
        Head.MillisecondsToday=Head.Hour.*3600000+Head.Minute.*60000+Head.Second.*1000+MillisecondsCurrentSecond;
        gJsf0080Write(JsfHead,Head,Data,[dName fName{nn}(1:end-4) '_' num2str(dSec,'%+9.3f') '.jsf'],0);disp('Mess 0080_1_022 completed');
        toc;
    end;
    clearvars dName fNameMap fId C fName LL ddSec nn dSec JsfHead Head Data Sd tmp1 tmp2 MillisecondsCurrentSecond tmp3 Dx Sd
end;
if strcmp(gKey{1},'InterpShift') %
    %{'InterpShift','c:\temp\000\interp.txt'};JsfTimestampShift;
    try fName=gKey{2};catch,fName=input('File for interpolation=');end
    %16/07/2023 21:12:41    GFO_0632_HDD_P195.jsf   -2.6
    fId=fopen(fName,'r');C=textscan(fId,'%f/%f/%f%f:%f:%f%s%f','Delimiter','\t','MultipleDelimsAsOne',0,'EndOfLine','\r\n');fclose(fId);
    GpsDay=gNavTime2Time('YMD32Dx',C{3},C{2},C{1});GpsTime=gNavTime2Time('HMS32Sd',C{4},C{5},C{6});
    F=struct('fName','','YY',C{3},'MM',C{2},'DD',C{1},'hh',C{4},'mm',C{5},'ss',C{6},'t',(GpsDay-GpsDay(1)).*3600.*24+(GpsTime-
GpsTime(1)),'GpsDay',GpsDay,'GpsTime',GpsTime,'LName',C(7),'Shift',C{8}','ShiftOut',zeros(size(C{8})));
    L=isnan(F.Shift);
    F.ShiftOut=interp1(F.t(~L),F.Shift(~L),F.t,'linear','extrap');
    %plShift=polyfit(F.t(~L),F.Shift(~L),1);F.ShiftOut=polyval(plShift,F.t);
    plot(F.t./3600,F.ShiftOut,'.-');hold on;plot(F.t(~L)./3600,F.Shift(~L),'*r');
    dlmwrite([fName '01'],F.ShiftOut','precision','%.3f','delimiter','\t','newline','pc');
    clearvars rootD dz fName nn LL fId C
end;

%mail@ge0mlib.com 03/08/2023
```